A Design and Implementation of a Balance Robot

Changchun Dong*, Jun Xia

Department of Communication and Information Engineering,Shanghai Technical Institute of Electronics & Information,Shanghai, China
dongchangchun@stiei.edu.cn, xiajun@stiei.edu.cn

Keywords: balance robot, sensor, PID control

Abstract: In this paper, a method of a robot balancing on a mechanical seesaw is studied. The project combines a lot of basic principles of robotics, sensor technology, bus systems, control algorithms and more. The robot takes C8051 F020 as the control chip, including the sensors of a gyroscope and an accelerometer. It uses C code with PID control to ensure that the robot can keep balance in a certain range. The result shows that every requirement can be implemented and works like intended.

1. Introduction

The goal of this project is to design and manufacture a robot that balances on a mechanical seesaw. If a counterweight is placed on the seesaw, the robot has to change its position to achieve a plane alignment. The robot has to react automatically to every change of position of the counterweight and keep the balance for a certain amount of time.

2. Method

2.1 Angle Measurement

Since the seesaw only produces a tilt of 11° angle this project has to be very accurate especially with small angles. To achieve this goal this project uses a combination of two sensor types: an Accelerometer and a Gyroscope.

The available Accelerometer is a type MXD2020E/FL from MEMSIC.

Figure 1 Breakoutboard of the Acclerometer

The signal the controller receives from the sensor is a 100 Hz Pulse Width Modulated (PWM) signal that represents the acceleration (MEMSIC,2003).

DOI: 10.23977/icidel.2018.001

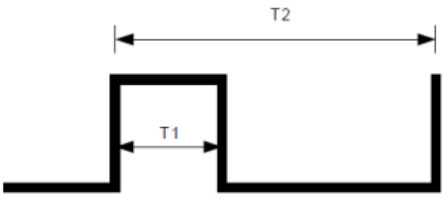

Figure 2 PWM Sketch

T2=10ms, T1 depends on the acceleration the sensor is exposed to.

If T1 is measured by the micro controller, the formula to calculate the acceleration in g is: $a_{(g)}$ = (T1/T2 − 0.5) / 0.2 withT1/T2 corresponds to the duty-cycle of the PWM signal.

To get accurate readings from the sensor, the micro controller has to measure two times, the High period and the Low period of the PWM signal. The High period time equals T1and High + Low period time equals T2.

For this purpose it uses one Programmable Counter Array module (Module 0) of the controller. Module 0 is configured to detect every transition of the signal (high->low and low->high) from the input on Port 1.0. When this happens, the controller produces an interrupt of the main program and vectors to the PCA_ISR (PCA interrupt service routine). With the help of a Flag (CCF0) it also follows the procedure to extract the High or Low period time.

Module 0 is also configured as a 16 bit counter, so the maximum value it can reach is 2^{16} = 65536. When this value is reached, the counter overflows and starts counting again from 0 to 65536. The PCA in general operates at a frequency of SYSCLK/4 which is exactly 5.529.600 Hz (~5.5 MHz). Since T2 is always exactly 10 ms, this time period can be used to decide if an overflow counter is required:

Time for 1 value: $1/5529600 \approx 1{,}808 * 10^{-7}$

Time for 1 overflow: $1{,}808* 10^{-7}* 65536 = 11{,}85ms$

One overflow takes more time than the maximum period time of the measured signal. This leads to the conclusion that it doesn't need to count overflows and can use the counter as direct reference for the times T1 and T2.

To extract an angle from the acceleration data of the sensor a few calculations have to be made.

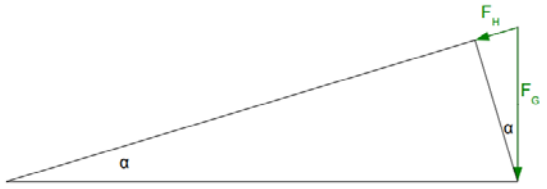

Figure 3 Sketch of the forces applied on the car when standing on the seesaw

Any text or material outside the aforementioned margins will not be printed.

The angle α: $\alpha = \sin\left(\frac{F_H}{F_G}\right) = \sin\left(\frac{m*a}{m*g}\right) = \sin\left(\frac{a}{g}\right)$

Since g is a constant, the only variable left is the acceleration a. To simplify calculations for the microcontroller and save processing time it also uses an approximation of this formula: α =a/g.

The approximation leads to accurate measurements between -30° and +30° angle. Everything below/above these angles should be calculated with a sinus function again. In our particular project this is not necessary.

A Gyroscope gives information of the orientation with the help of angular velocity. This means the sensor measures the actual speed it is rotating around an axis. This angular velocity can be multiplied by the time, so it is possible to calculate the angle it changed. This project uses a L3GD20 from STMicroelectronics (STMicroelectronics, 2013).

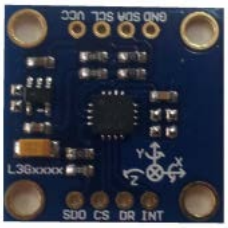

Figure 4 Gyroscope L3GD20 in Breakout Board

It used the communication interface SPI Bus to extract the Raw Data from the sensor. It uses 5 Pins on the microcontroller:

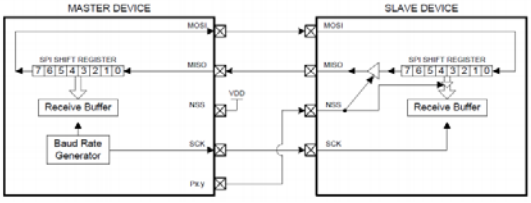

Figure 5 Sketch of SPI Bus Communication with 1 Slave and Full Duplex Operation

The output registers mentioned earlier are transmitted as 16 Bit values for each axis measured. They represent the angular velocity in a certain time period. The angular velocity can then be calculated with this formula: angular velocity = (250dps /65536)∗ value.

Integration of angular velocity leads to an angle in degrees(Yafei Ren,2007).

$$\text{angle}_{(\text{degree})} = \int \text{angular velocity} * dt$$

2.2 MCU

The used MCU belongs to Freescales 8-Bit family and uses the popular 8051 architecture, it offers several on-chip benefits such as different timers, ADCs/DACs and bus-interfaces (I²C, SPI).

The C8051 F020 is a good solution for the given task, as the controller is able to connect to many different sensor types like analog sensors, digital sensors, sensors with PWM outputs or bus-connected sensors via I²C or SPI(Freescale Semiconductor,2003).

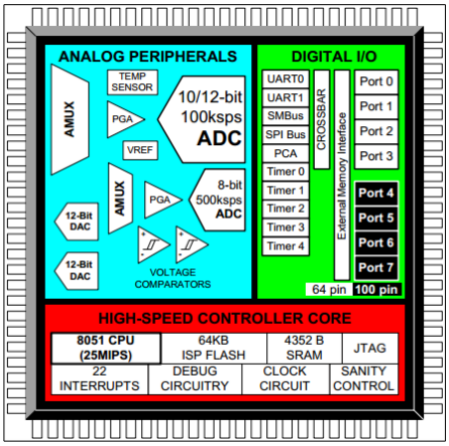

Figure 6 Block diagram of MCU

2.3 Procedure

In a normal program sequence every chain gets the same percentage of power (duty-cycle of PWM) either from the control algorithm or from a fixed value. This percentage is given to an output program that directly generates the PWM signal for the motor driver. The only way to steer the car is to alter one individual chain power separated from the other one. This is achieved by adding an additional program between these two programs:

// PD Algorithm pwm_l = pwm_r = control_algorithm ();
// steering pwm_l += steer (pwm_l, left, 10);
pwm_r += steer (pwm_r, right, 10);
// Output Controller_Output (pwm_l, pwm_r);

By constantly calling this sequence the steering can effect the output always as needed.

A closer look at steer ():

• Declaration: char steer (char PWM, bit side, char offset);

• Input values:

1) PWM – The desired acceleration in %

2) side – This bit determines the chain that has to be checked for steering

3) offset – This value adjusts the impact of the steering. A higher value leads to stronger steering until the car rotates on its position (offset = -PWM)

• The Return value is 0 when the opposite sensor detects no reflection and (+/-) offset if the opposite sensor detects a reflection. The sign depends on the direction the car is moving (positive or negative PWM).

2.4 Control Algorithm

The control algorithm that places the robot in the right position on the seesaw can be altered between a PD or PID algorithm (Shousong Hu, 2013). Both algorithms have their advantages and disadvantages. To get a better understanding of this the project describes every element of the algorithm for its own.

The main goal of the algorithm is to set an appropriate output value for the motor. The formula for this is very simple:

output = Kp * p + Ki * i + Kd * d

p stands for proportional and contains the difference between a certain set point (normally 0) and the actual angle of the car. (p = set point – angle)

i stands for integral and contains the summation of differences between a set point and the actual angle of the car. If the car stands for 3 periods in a 5° angle (with set point = 0), i will be 15. The i part grows bigger and bigger over time if an error stays in the system.

d stands for derivative and is higher than 0 if the seesaw is rotating. The d part also counteracts the actual movement from the other part of the algorithm. The d part is the “break” of the system and without it the car will never be able to stop at the right time on the seesaw.

3. Result &Conclusions

Every requirement can be implemented and works like intended. The project combines a lot of basic principles of robotics, sensor technology, bus systems, control algorithms and more. The different control modes could easily demonstrate the difference between P, PD and PID control systems. The angle measurement includes different communication systems with electrical elements, filter technology and the digital implementation of sensors in general. The line tracking describes a

simple way to navigate robots in their surroundings. Also the whole code is written in C program language and could be used as an example of how the language is used in robots.

References

[1] Freescale Semiconductor, 2003, C8051F02x.
[2] MEMSIC, 2003. MXD2020E/FL.
[3] Shousong Hu, 2013. Principle of Automatic Control (in Chinese), The Science Publishing Company.Beijing, 6nd edition.
[4] STMicroelectronics, 2013. L3GD20.
[5] Yafei Ren, Xizheng Ke, Yijie Liu, 2007. MEMS Gyroscope Performance Estimate Based on Allan Variance. In Proceedings of 2007 8th International Conference on Electronic Measurement & Instruments.Vol.1, pp260-263.